EDMONTON
99'er
COMPUTER
USERS'
SOCIETY

99'er Online

NOVEMBER 1984 EDITION

P.O.Box 11983
Edmonton, Alberta
Canada T5J 3L1

TO:

99'er ON LINE is the news letter of the Edmonton 99'er Computer User's Society published ten times a year. All material contained in this news letter may be published in other news letters provided that source and author are identified unless otherwise stated. We welcome correspondence from all TI User Groups and will extend source credit courtesy in **99'er ON LINE.**

CORRESPONDENCE: News letter editor: BOB PASS, 59 LABELLE CR, ST. ALBERT, ALBERTA, CANADA T8N-2G6. All other correspondence: EDMONTON 99'er COMPUTER USER'S SOCIETY, P/O BOX 11983, EDMONTON, ALBERTA, CANADA T5J-3L1

OFFICERS: PRESIDENT--BILL CANNON, VICE PRES--PAUL HELWIG, TREASURER--EVAN SMITH, SECRETARY--SUSAN LIVINGSTON

DISCLAIMER: All information published in this news letter is, for the most part, the fruits of the labors of amateurs; therefore, we cannot guarantee that the information presented is always correct.

REGULAR MEETINGS: Regular meetings of the Edmonton User's Group are held on the second Tuesday of each month on the 3'th floor of the General Services building of the University of Alberta from 7:00 till 10:00 PM and are open to all members in good standing. Non-members may attend their first meeting free of charge. The Executive Committee meets monthly. Members may attend these meetings as observers or to address a particular issue. Arrange with one of the officers listed above if you wish to attend.

ADVERTISING: Commercial advertising space is available in this news letter at the following rates: FULL PAGE--$20.00, HALF PAGE--$15.00, 1/4 PAGE--$10.00. Discuss your commercial needs with Paul Helwig at the next meeting or write to the P/O Box above. Members may advertise their personal computer related items for free but are asked to limit their ads to about 20 words. Mail your ads to the EDITOR'S ADDRESS or hand it to him at the general meeting; newsletter deadline 15'th of the month.

MEMBERSHIP FEES: FAMILLY---12 MONTHS, $20.00, 6 MONTHS, $15.00. STUDENTS---12MONTHS,$15.00, 6 MONTHS, $10.00.

THE EDMONTON 99'er COMPUTER USERS' SOCIETY

Date   /   /

New Membership Application ☐

Renewal of Membership ☐

Change of Address ☐

NAME ____
last   first   initial

ADDRESS ____

CITY ____________________ PROVINCE or STATE ____________________

POSTAL or ZIP CODE ______________ TELEPHONE ____________________

Single or Family membership	$20	(12 months)	☐
	$15	(6 months)	☐
Student membership (must have student I.D. number)	$15	(12 months)	☐
	$10	(6 months)	☐
Out of Town (newsletter only)	$20	(12 months)	☐
	$15	(6 months)	☐

* Please note that all membership money should be in CANADIAN FUNDS.

* May we release your name, address and telephone number on a club roster, to companies and other clubs?

YES ☐   NO ☐

FOR OFFICE USE ONLY

Date Filed   /   /   Membership Number ____________________

Amount Tendered $ __________ Authorized Signature ____________________

DISK COPYING IN TI-FORTH

by: Michal Jaegermann

Everyone who has tried to run FORTH for a while knows perfectly well that such activity involves a lot of disk copying. Normally you have a "master copy", it's backup, your "work copy", it's backup, and other disks where you created bigger aplications in binary saved (BSAVEd) form - possible with some overlays - and so on. Even if you are not adhering to such a policy of backups, you should reconsider - it is extremely easy to trash a disk using FORTH. According to the philosophy of the FORTH language, the whole responsibility for error control lies with the programmer. Therefore FORTH does NOT have all the nice safeguards which prevent silly actions. This is really great since you have access to otherwise unavailable features of your machine - but only as long as you are absolutely sure of what you are doing. Morever, the system is not guaranteed to be a bug free; and indeed it isn't as you will see later in this article. So far, I am not aware of anything really devastating in TI FORTH and up to this date everything runs fine, but one should take this into account when trying new features. Therefore, try your bright new ideas on backups of backups!

This policy is quite easy to implement when you have two disk drives. The "BLOCK DROP UPDATE" method described in the TI-FORTH manual is useful if you have only a couple of screens to copy. Your system disk can be copied using the Disk Manager; consult the section "DISK UTILITIES" in chapter 5 of the manual. One little twist: the method will work fine on standard single sided TI disk drives but if you happen to have double sided drives then the procedure is longer. First, initialize a blank disk as two sided in order to make it accessible by FORTH later. "FORMAT-DISK" cannot do that; use the Disk Manager. Do not try to copy a FORTH disk in this mode since the Disk Manager will hash it's contents and it will make your copy quite useless. Next, reinitialize your disk as single sided and copy. Do not pay any attention to the disk catalog as returned by the Disk Manger as this information for FORTH disks is absolutely worthless. Do not attempt to put any files onto the FORTH disk (system or non-system) even if the Disk Manager is indicating that some sectors are free.

Still sounds troublesome? Well, one drive blues can be cured, at least to some extent, by the following TRANS utility. This word copies FORTH disks with one drive and up to 30 screens in one swap instead of the 4 screens in the resident "FORTH-COPY" routine. By default, this utility copies a whole disk; but if you type "27 51 FROM/TO" then TRANS will pick up only screens from 27 to 51 (inclusive) and will deposit them on your copy disk as a consecutive block starting at screen 27. The same effect will be achieved if you execute "27 25 FROM/CNT", since you are copying 25 screens. If you would like to write those screens to a disk starting at screen 43, execute "43 TARGET" and TRANS will dutifuly deposit your screens starting at 43 and ending with 67. If you try to use values which would require access to non-existing sectors, then your request will be politely ignored and you will be informed about it. This is not absolutely foolproof since error trapping depends on a value stored in a user variable, "DISK_SIZE". The default is 90, which corresponds to a standard TI single sided drive. For two sided drives, change "DISK_SIZE" to 180; also reset "DISK_HI" accordingly. Improper values in DISK_SIZE may cause an error. Please note that error messages are very limited ones, so when you see "Disk Error" on your screen it likely means that there is a disk access error, not neccessarily a fault in TRANS.

TRANS is reusable. This means that after a succesfull transfer, error, or when you Q(uit) you can pick up the next block of screens or the next disk and execute the whole procedure again. Type TL<ENTER> to get the title screen with information on the current settings of parameters.

Please note that in order to make space for copying buffers, all words which do not belong to the FORTH core were "forgotten" when TRANS was loaded. All five RAM disk buffers are needed so if you have installed something in that area, unpatch it and restore 5 buffers. Also VDP memory is rearranged: on line #1 of the first screen, you wil find a routine which is equivalent to CALL FILES(1) in Basic. Moreover, the internal FORTH disk buffer is shifted down into a pattern table for ASCII characters 128 to 255. So when you are through, it is neccessary to re-boot your FORTH system. The simplest way to do that is by typing COLD<ENTER>. This will reload your standard screens. Please note that although DISK_BUF will be shifted bak to >1000, you will still have only one file buffer (this will matter only if you are about to use a routine which requires more than one file opened) and part of the pattern table will contain garbage left from the last disk access. The word TEXT, (if it's loaded) will fix this up or perhaps you could write your own cleanup procedure. Of course, TRANS has been forgotten in the process. The next time you want to use TRANS, re-LOAD the corresponding screens once again.

One final word of caution. Do not use TRANS to copy screens inside one disk if you are moving more than 30 screens and ranges overlap as you may loose some screens; otherwise this practice is safe.

```
( FAST COPY ver.3 - 1st scr. * Michal Jaegermann )  BASE->R HEX
PABS @ 1 OVER VSBW 16 OVER 1+ VSBW 1 834C C! 8356 ! A E SYSTEM
C00 DISK_BUF ! 0 DISK_LO ! ' TASK LFA @ FENCE ! FORGET TASK
1000 VARIABLE VHR          0 VARIABLE WHR        1414 CONSTANT VOL
: DS DISK_SIZE @ ; DS VARIABLE CNT 0 VARIABLE BGN
0  VARIABLE SFT           : AT GOTOXY ;       : CS F4 7 8 SYSTEM ;
: CLS 10 SYSTEM ;         : VMBW  2 SYSTEM ;    : VMBR 6 SYSTEM ;
: W CLS D C AT ." WRITING DATA..." FLUSH ;
: M1 CLS 4 C AT ." READING SCREENS..." CR CR ;
: M2 A 14 AT ." PRESS S TO START" 10 16 AT ." Q TO QUIT" ;
: MSGW CLS F7 7 8 SYSTEM A 8 AT ." INSERT COPY DISK"   M2 ;
: MSGR CLS FD 7 8 SYSTEM A 8 AT ." INSERT SOURCE DISK" M2 ;
: WAIT BEGIN KEY DUP 51 = IF CLS 0 0 AT CS -3C3C ALLOT ABORT
     ENDIF 53 = UNTIL ;
: ERR CLS CR CS F000 HERE U< IF ." Disk error" -3C3C ALLOT
  ELSE ." Error" ENDIF CR ABORT ;                    -->

( FAST COPY ver.3 - 2nd screen * Michal Jaegermann )
: QRM ( scn n1 n2 -- m1 scn [m2] f )
 >R 1- R> /MOD >R 1+ SWAP R> -DUP ;
: BLRD ( scn1 n -- scn2 ) EMPTY-BUFFERS
0 DO DUP 4 .R DUP SFT @ + OVER BLOCK 2- ! UPDATE 1+ LOOP ;
( reads up to 5 scns starting with scn1, first unread scn2 )
: PUSH 3 0 DO 5 BLRD VOL WHR FIRST OVER @ 2- MOVE +! LOOP ;
: PUSHV -DUP IF 0 DO 5 BLRD VOL VHR FIRST OVER @ VOL VMBW +!
  LOOP ENDIF ;
: POP 3 0 DO WHR @ VOL - DUP FIRST A0A MOVE WHR ! FLUSH LOOP ;
: POPV -DUP IF 0 DO VHR @ VOL - DUP FIRST VOL VMBR VHR ! FLUSH
  LOOP ENDIF ;
: BLMV ( scn1 n -- scn2 ) MSGR WAIT M1 5 QRM DUP >R
   IF 3 /MOD IF SWAP PUSH SWAP ENDIF PUSHV ENDIF SWAP BLRD
  MSGW WAIT W R> -DUP IF 3 /MOD IF POP ENDIF POPV ENDIF ;  -->
( copies up to 30 scns - begin scn1, first uncopied scn2 )

( FAST COPY ver.3 - 3rd screen * Michal Jaegermann )
: TL CLS 8 3 AT ." FAST COPY - VERSION 3.0" 3 A AT ." Current va
lues for BGN and COUNT" 3 C AT ." are " BGN @ DUP . ." and " CNT
@ . ." Image starts at" 3 E AT ." screen # " SFT @ + . 16 E
AT ." To change use" 4 10 A." FROM/CNT or FROM/TO and TARGET"
3 15 AT ." Execute TRANS to start copying" SP! QUIT ;

: CHK OVER OVER 1- + DS < OVER 0 > AND ; ( n1 n2 -- n1 n2 f )
: TARGET CNT @ CHK IF DROP BGN @ - ELSE 0 ENDIF SFT ! TL ;
: FROM CNT CHK   IF CNT ! BGN ! ENDIF TARGET ;
: FROM TO 1+ OVER - FROM/CNT ;

: TRANS CLS 1000 VHR ! HERE WHR ! 3C3C DUP ALLOT BGN @ CNT @
  1E QRM IF 0 DO 1E BLMV LOOP ENDIF SWAP BLMV DROP CS CLS
  8 C AT ." **** DONE ****" MINUS ALLOT  TL ;
R->BASE    -1 WARNING ! ' ERR CFA ' (ABORT) !                 TL
```

Now for some tips for those with at least two drives who wish to use TRANS. S imply type DR1 after TRANS is loaded. FORTH will then read screens from drive 1 and write them to drive 0.

You can also remove WAIT from BLMV. SOme simpler methods are also available. Namely, the word -COPY loads screen 39 from the system disk on which the word FORTH-COPY is provided. It will copy a disk in drive 1 to a disk in drive 0 as it now stands. The only problem is that the word is slow and, which is worse, it contains a bug. If you happen to have two-sided drives, it will copy the second half of the disk in drive 0 onto first half of the same disk. Similar effects occur for a double density drive. An analogous bug. but of lesser consequnce, can be found in the word DTEST. Therefore, my advice is to retype screen 39 of your system disk as shown below. This version of FORTH-COPY is longer, but it will take into account your declared DISK_SIZE, will read and write 5 screens at a time, and will leave you, after execution, on the same drive where you were when you typed FORTH-COPY<ENTER>. **NOTE:** It still copies from drive 1 to 0. Do not forget to store 0 in DISK_LO; otherwise you will endup with a disk access error on the first attempt to write.

This version would still work if your buffers were smaller than five screens, but not as effectively. You can change from 5 to your buffer size. This number can be placed on the stack by the following sequence: LIMIT$ @ FIRST - B/BUFF$ @ 4 + / which changes the buffer to 4 for example and can be placed directly into your definition of FORTH-COPY.. However, if that number does not divide DISK_SIZE evenly, then some screens will not be copied. You could extend your definition of FORTH-COPY to take that into account. A simpler solution with, for example, 90 DISK_SIZE and 4 screen buffers would be to use a constant of 3 instead of 5.in the colon definition of FORTH-COPY. It will be actually faster in that case than the one with an original 5.

```
( STRING STORE AND SCREEN COPY WORDS 10SEP84 )  0 CONSTANT AD
 0 CLOAD FORTH-COPY                            BASE->R  HEX
: (!") R COUNT DUP 1+ =CELLS R> + >R >R SWAP R> CMOVE ;
: !" 22 STATE @ ( store string at addr ) IF COMPILE (!")
WORD HERE C@ 1+ =CELLS ALLOT ELSE WORD HERE COUNT >R SWAP R>
CMOVE ENDIF ; IMMEDIATE DECIMAL : DTEST DISK_SIZE @ 0 DO I
DUP . BLOCK DROP LOOP ; ( screen copying words ) : SCOPY
OFFSET @ + SWAP BLOCK 2- ! UPDATE FLUSH ; ( 1K blocks ) :
SMOVE >R OVER OVER - DUP 0< SWAP R MINUS > + 2 = IF OVER
OVER SWAP R + 1- SWAP R + 1- -1 ' AD ! ELSE 1 ' AD ! ENDIF
R> 0 DO OVER OVER SCOPY AD + SWAP AD + SWAP LOOP DROP DROP ;
: FORTH-COPY EMPTY-BUFFERS OFFSET @ DR1 DISK_SIZE @ 0 DO CR
5 DUP I + I DO I DUP DUP 6 .R BLOCK 2- ! UPDATE LOOP FLUSH
PAUSE IF LEAVE ENDIF +LOOP OFFSET ! ; R->BASE
```

Please note that word "-->" was removed from the last line of this screen. The next screen contains only a colon definition of DISK-HEAD and I cannnot find any good reason to load that word any time I want to do some screen copying. Actually words (!") and !" are also not necessary for that purpose, but they are often valuable on their own (you have to have them before you can load DISK-HEAD). But how you organize your screens and what you will choose to load with every option is entirely your choice!

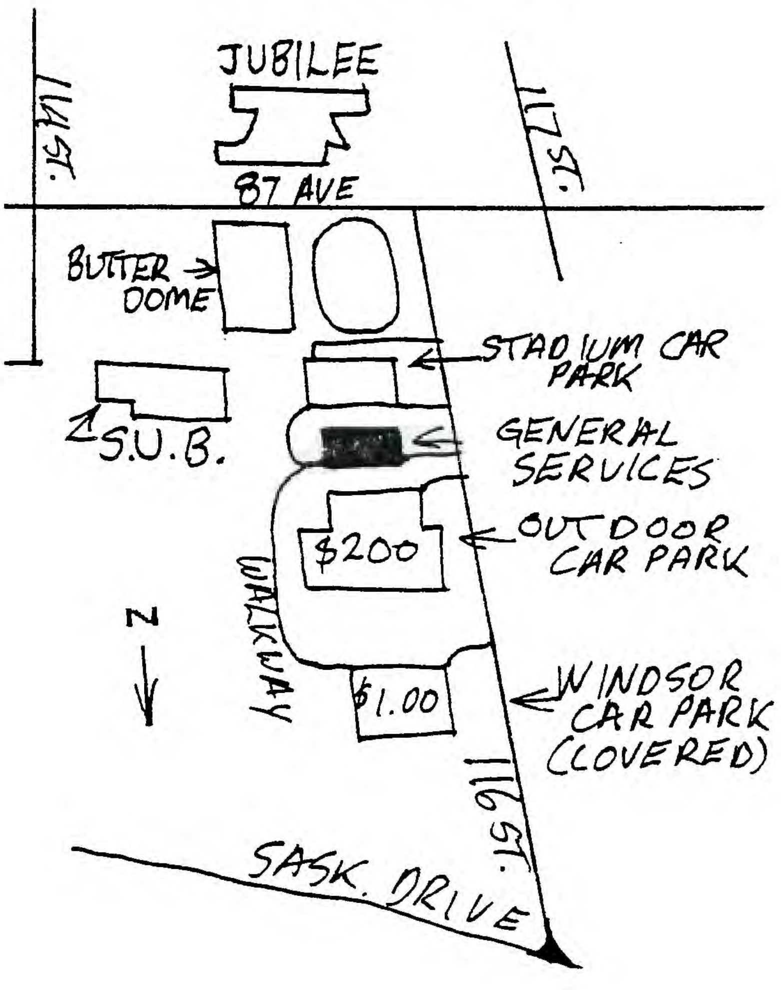
A CALL KEY CURSOR

by: Pradeep Bhatia

Your CALL KEY command, unlike the INPUT command, does not display a cursor while scanning for a key stroke. The following consol BASIC program displays a flashing cursor while waiting for you to press a key. Also, it displays the key character you have pressed just like INPUT does. Try using this idea to avoid error messages generated by an INPUT command which receives a non-numeric character into a numeric variable.

```
100 CALL SCREEN(8)
110 CALL CLEAR
120 PRINT "CONTINUE? (Y OR N
)"
130 CALL HCHAR(23,22,127)
140 FOR X=1 TO 5
150 NEXT X
160 CALL CHAR(127,"000000000
0000070")
170 CALL KEY(3,K,S)
180 IF K=89 THEN 190 ELSE 23
0
190 CALL HCHAR(23,22,89)
200 FOR X=1 TO 100
210 NEXT X
220 GOTO 130
230 IF K=78 THEN 240 ELSE 26
0
240 CALL HCHAR(23,22,78)
250 STOP
260 FOR X=1 TO 50
270 NEXT X
280 CALL CHAR(127,"000000000
0000000")
290 CALL KEY(3,K,S)
300 IF K=89 THEN 310 ELSE 13
0
310 CALL HCHAR(23,22,89)
320 FOR X=1 TO 100
330 NEXT X
340 GOTO 130
350 END
```

University of Alberta



campus map

*** IMPORTANT ANNOUNCEMENT ***

GUEST SPEAKER

At our November meeting we will be privileged to have as a guest speaker Mr. Martin Kratz from the University of Alberta. Martin acts as a consultant to law firms, corporations and government in the development of protection strategies for high technology developments with a particular emphasis on the protection of computers, computer programs, and data. Martin is presently completing his articles with the Alberta Superior Courts, and lectures at N.A.I.T. and at the University of Alberta on issues of computer law.

Martin will be speaking on the various aspects of ethics as it relates to the field of computers and computer software. He will touch upon such topics as the warranty disclaimer and what it means, the implications of software piracy and the security of computer systems in general.

FOR SALE

A complete TI system! PE Box, RS232, 32K Mem. exp, Disk Drive and Controler, X-Basic, software. Excellent cond. Call Graeme Horne - 484-3719.

BASIC PROGRAMMING

by: Bob Pass

Last month's article was on variables of the numeric kind and why we used them in computer prgramming. This month, as promised, I will discuss string variables. Once again, before you proceed, review in your User's Reference Guide the conventions to be used in naming variables: see section II pages 11, 15, and 16. Pages 45, 58-60, and 61-64 show some good examples of how both types of variables are used.

The most important thing to remember when using strings is that the variable name must end with a dollar sign ($) so that the machine will know that you are not using a numeric variable. Why is this important? Well, when you define a numeric variable, the computer knows exactly what to expect; numbers are clearly defined and are limited in size. In fact, any number you may want to work with can be stored in memory in just 8 bytes (that's computerese for those "pigeon holes" or memory addresses I spoke of last month)! Therefore, when you define a numeric variable, the machine reserves an 8 byte block of memory space for the storage of that number. (At this point, if I have lost you, please review last month's article). Strings on the other hand, can contain any of the 255 keyboard characters you can type and, moreover, can be from 0 to 255 characters long! In fact, that's why they are called strings because they are visualised as a line of characters strung out like beads on a string. It should be obvious that because strings can be so long it makes good sense from a memory usage point of view to use a variable name rather than repeating a string several times in a program.

Now, let's look at this from the machine's point of view when you assign a string variable. First of all, the machine knows full well that if you enter A$="BOO" that it will need a fixed amount of memory to store that information. It also knows, however, that you could later re-assign that varible name to something else, like A$="[illegible]-HOO", which requires more memory space. The computer just does not know what to expect next with strings! It could, of course, reserve a block of 255 bytes for each string variable and treat the strings just like numbers but that would eat up memory space very quickly. Since strings are by nature dynamic, the machine must use a dynamic, or changeable, system to administer the space required to store strings.

How is this done? First of all, when the computer scans your program after you have typed RUN, it enters each string variable name in a table just like it does with numbers. Secondly, from the pre-scan, it knows your initial definition and length of the string (A$="BOO" is 3 characters) so it can enter in the table a pointer (address) to the block containing the string definition. The block will need to be 4 bytes long for the example shown here; the first byte indicates how many characters are in the string and the following bytes contain the ASCII codes of each character in the string using one byte per character. (The machine does not actually store the letters you type but rather the 8 bit code numbers that represent them. These code numbers are called ASCII codes). If your definition is A$="" (this is a "NULL" or the equivalent of zero as there are no characters between the quotes) then the block would be one byte long and would contain 0 indicating no characters. The pre-scan will alot a NULL to every string that will be assigned with an INPUT statement. (By the way, remember that the space character is not nothing! A$=" " is not the same as A$=""). Since each byte in memory can contain 8 bits of binary data and we are restricted to one byte to indicate the length of the string, then the maximum length of a string is 255 characters as this is the largest number which can be expressed in 8 bits (2 to the 8'th). Now, let's say that as the program is executing, A$ is redefined: A$="Boo Hoo" for example. This will not fit into the space presently reserved for A$ as it is now 7 characters long, not 3. (Count 'em again, there's 7 there)! The machine must now exercise it's dynamic powers of memory usage. First, it goes to the old block and zeros it. Next, it enters the address of the start of free memory space into the variable table entry for A$ (the computer always keeps track of where the unused portion of memory is located). Third, it writes the 8 bytes required starting at that address to define the new string and finally it updates a pointer to the free space address. Notice that this has left a 4 byte empty block from the old definition in memory. If the machine runs out of free space, it will initiate a routine to scan through memory and repack it to get rid of those empty holes thus freeing space at the bottom of the stack to be reused. Of course, the variable tables are also updated during this repack.

I assumed that you were all thoroughly familiar with numeric operators (+, -, *, /, etc) when I discussed numeric variables last month, so I didn't spend any time on them except for a description of what happened in memory when two numbers were added. Many of you are probably not familiar with string operators so here's a quick summary of the commands to show what you can do with strings. Please refer to the User's Reference Guide for each command to get a full explanation.

& The "&" symbol is the CONCATENATION operator which joins two strings together. If A$="BOO" and B$="HOO" then the statement C$=A$&B$ would make C$ equal to "[illegible]". If you wanted a space between the words then C$=A$&" "&B$ would do the job.

ASC This command will produce the ASCII character code number of the first character in the string specified.

CHR$ This command will convert an ASCII code number into a character. Very usefull for characters that cannot be printed.

LEN Produces a number equal to the LENgth of the string; ie, the number of characters in the string including spaces.

POS Will produce a number which indicates the POSition of the start of a specified string imbedded within a larger string and you can specify where to start searching in the large string.

SEG$ Allows you to extract a SEGment of a string from a specified string. You specify the starting position in the string and the number of characters to extract.

STR$ Converts a numeric value or variable into a STRing variable.

VAL Changes a string containing numbers only (and sign if required) into a numeric variable or VALue.

The following is a short program that illustrates some of these inctions. See if you can figure out what will happen before you JN it. Next month, I will cover graphics and I'll include a ogram to cover all three lessons.

```
100 CALL CLEAR
110 PRINT "CONCATENATION"::::
120 INPUT "YOUR FIRST NAME? ":N1$
130 INPUT "YOUR LAST NAME? ":N2$
140 NAME$=N1$" "&N2$
150 PRINT ::"YOUR CONCATENATED NAME IS:"::NAME$::::
160 N=POS(NAME$," ",1)
170 PRINT "THE SPACE CHARACTER IS IN POSITION #";N;" OF YOUR
NAME."::
180 L=LEN(NAME$)-1
190 P1=LEN(SEG$(NAME$,1,N-1))
200 P2=LEN(SEG$(NAME$,N+1,L-P1))
210 PRINT "THERE ARE";L;" LETTERS IN":"YOUR NAME:":P1;" IN
THE FIRST,":P2;" IN T HE LAST."::::
220 GOSUB 360
230 PRINT "THE ASCII CODES FOR YOUR NAME ARE AS FOLLOWS
(INCLUDES THE SPACE):"::::
240 FOR X=1 TO LEN(NAME$)
250 L=ASC(SEG$(NAME$,X,1))
260 PRINT L;
270 NEXT X
280 GOSUB 360
290 PRINT "INPUT A NUMBER BETWEEN 32  127 TO SEE WHAT
CHARACTER IT IS."::"TYPE A 1 TO STOP."::
300 INPUT N
310 IF N=1 THEN 350
320 IF (N>32)*(N<127)THEN 330 ELSE 300
330 PRINT "CHAR= ";CHR$(N)::
340 GOTO 300
350 END
360 PRINT ::"PRESS A KEY PLEASE"
370 CALL KEY(0,K,S)
380 IF S=0 THEN 370
390 CALL CLEAR
400 RETURN
```

REDFORD Computers

Canada's Largest Stock of 99/4A Hardware and Software. Many Third party 99/4A and other computer products available.

-Extended Basic $139.95
-Axiom Interface $149.95
-Super Extened Basic $129.95
-Panasonic KXP 1090 Printer $399.95
-MBX incl. Baseball & 1 other game $199.95

14648-134 avenue
Edmonton, Alberta.
Canada T5L 4TA
403-451-4529

...AND SORTING ONCE MORE

by: Michal Jaegermann

In the June issue of "99'er On Line". Bob Pass showed how to program "bubble sort" in BASIC. To be sure, there are a lot of different ways and methods to do the job. The 3'rd volume of "Art of Computer Programming" by D. E. Knuth (kind of a professional programmer's bible) is entitled "Searching and Sorting". Around 380 pages from this book are devoted to sorting but do not exhaust the subject. Obviously, we are not going to reprint all of this in our bulletin!

Nevertheless methods do exist which can be programmed in BASIC as easily as "Bubble Sort" and which work MUCH faster than it. In this note I would like to present to you the "Shell Sort" routine. Just for a change. I'll deal with string arrays. but they can easily be converted to sort arrays of numbers. Try them with arrays with around 500 elements in them to see how effectively they work. It should take around 4 minutes of sorting time. I would never make such a recommendation for a "Bubble Sort" (still you can try it once - just for a comparison purposes) ; on our machine a "Bubble Sort" of 150 elements is little bit on the joking side.

Just for the record. the name of the method does not come from some mysterious shell. but from the surname of a fellow who first decribed that kind of sorting. D. L. Shell.

Below is the program. It sorts strings in a "lexicographical order". which means - like in a dictionary. Of course. before you will use it you have to declare and initialize your arrays. The simplest way to get them is to generate a lot of random numbers and then convert them into strings using the STR$(---) function in BASIC. Just remember that in this arrangement 2 is greater (i.e. folows as a string) than 198784728. (Also. the sort will be by ASCII code value which can look pretty mysterious at times! -ED).
For a start here is an EXTENDED BASIC version. (Number N is a dimension of an array to be sorted. OPTION BASE 0 is assumed. "*" is an asterisk).

```
100 GAP=3*N/2
110 IF GAP=1 THEN 150 ELSE
    GAP=INT(GAP/2)
120 FOR I=0 TO N-GAP :: L=I
    K= L+GAP :: T$=A$(K)
130 IF A$(L)>T$ THEN A$(K)=A
    :: A$(L)=T$ :: IF L>GAP
    K=L :: L=L-GAP :: GOTO 1
140 NEXT I :: GOTO 110
150 PRINT "DONE"
```

Now for all of you who have only console BASIC. a simple translation into that language:

```
100 GAP=3*N/2
110 GAP=INT(GAP/2)
120 FOR I=0 TO N-GAP
130 L=I
140 K=L+GAP
150 T$=A$(K)
160 IF A$(L)<=T$ THEN 230
170 A$(K)=A$(L)
180 A$(L)=T$
190 IF L<=GAP THEN 230
200 K=L
210 L=L-GAP
220 GOTO 160
230 NEXT I
240 IF GAP>1 THEN 110
250 PRINT "DONE"
```

Now try it to see how it works. seeing is believing! How does this method do that? Quite simple. When it is circulating through the array it looks at pairs which are "GAP" apart. If they are in the wrong order, it swaps them (line 130 in X-BASIC and lines 160-180 in BASIC program). After every such exchange, the program backs up, if possible, in order to check that the previous pair was not affected. If this is the case, the whole cycle of swaps and checks is restarted once again (GOTO 130 in XB and GOTO 160 in BASIC). In fact this is the Bubble Sort, but performed only on a sequence of elements "GAP" apart. A FOR...NEXT loop repeats that cycle with every available starting point. On the next iteration of the program, "GAP" is halved and the job is repeated once again. When we do that with GAP=1 then, of course, everything is in a proper order since every two consecutive elements were checked and reordered when necessary. Halving is a really fast way to reach 1. You have to do that only ten times when starting with 1024. An inital value of GAP is only 3*N/4 (there are some reasons to choose that formula, but they are not very important to us). Notice also that on every iteration more and more pairs are ordered, hence the chances that we will be able to skip many swaps are growing pretty fast. So, "Shell Sort" next time!... (unless you write an Assembler routine which will do the job much faster - but even then "Shell Sort" is a very good option to use in Assembler!).

FOR SALE

DIG DUG cartridge for sale. $20.00 - call Paul @ 475-1439.

THE DUNGEONS OF KRUNG GAME REVIEW

by: Francis X. Gaston

This is one of the best "screen" type games that I have ever seen written in console basic. It is written by Jim Beck, a 12 year old, from Edmonton, Alberta, and is available under 4A Software, from the Games Gang. Five other titles are also available from 4A Software. System requirements for this game is a console, joysticks, cassette recorder, or disk drive.

This is a screen driven game (5 screens), much like you would see for Donkey Kong or Miner 2049'er. The objective is very simple, to achieve freedom from the dungeons of the planet Krung. You may run, jump, or use your jet pack to gain your objective. You must avoid the destructive and explosive white cubes while trying to stay on the red platforms or telepost tubes. You may collect treasuress on your journey to achieve higher bonus points. Clearance of the screen is achieved when you encounterthe checkered cube on the top of the screen. Once you reach the surface of the planet though, you must still navigate through a mine field. Once this is done, you are returned to screen one where you start guiding another prisoner to freedom.

The game is excellent considering it is written in basic. It is extremely fast and the graphics are unmatched in it's class. It is also very addictive; your objective of reaching screen 5 appears foremost on your list. Since you are allocated only one prisoner per game, one wrong step is an immediate termination of the game. You will have to restart at the very beginning once again. This at times can be frustrating but patience is truly a virtue. Since there is no time limit, it is sometimes best to analyse your next steps, or else ----- KA-BOOM!!!

editor's note: Francis is our one and only member living in Saskatoon. He recently sent me several good articles for the newsletter which will be published in the following months. Francis may not realise (and most of you as well) that Jim Beck is also a member of our group who comes to most meetings. I'm sure Jim will be pleased to hear how well his product is received.

LITTLE GEMS

If you have ever tried to produce a solid horizontal line but the best you could get was a broken dotted line try this:

```
CALL CHAR(95,00FF)
```

This changes the underline character (FCTN U) to be a solid line when more than one character is typed. To change the position of the line vertically, add or remove pairs of zeros; to make the line heavier, add pairs of F's. For a fine dotted line, replace the F's with A's.

NOTES FROM THE EXECUTIVE

THANKS: A note of thanks to Marlene Jowell of Wordware Publishing, Plano, Texas. Marlene sent us a free copy of "Learning TI-99/4A Home Computer Assembly Language Programming" for our perusal. It looks like a darn good book; if you would like a copy it is available at Audrey's Books on Jasper Ave for $22.95. It will be available for your viewing pleasure at the next meeting.

NEXT MEETING: Will be held on November 13'th @ 7:00 PM sharp. There will be a guest speaker (see article) so please try to get in before the meeting starts. Doors open @ 6:45 in room 821 (or 849), 8'th floor of the General Services building on U of A campus (see map). Signs will be posted in the hall to guide first timers to the room.

TELEPHONE ANSWERING LINE: An automatic telephone answering machine and dedicated telephone number for our members to use in contacting the executive was researched but was found to be too expensive at this time. If we get an increase in new members (ie sufficient bucks), this service will be reconsidered.

BASIC PROGRAMMING COURSE: Our trusty scribe, Bob Pass, will conduct a course in TI Basic programming. This course is intended for those who need Basic basics only and will be held in the students homes on a "share the host" basis. Bob wants to keep the class size to three students to increase interaction. The first session of four lessons is full; the next session will be in January so see Bob to register for it.

DISKETTES: Gord Bradlee will be obtaining a supply of diskettes at $2.00 each. Place your order at the next meeting and he will bring them to the following one. Cash up front please.

MEMBERSHIP CARDS: If things went as planned, there should be a membership card included with this copy of your newsletter; if there wasn't, look for it next month. Please bring the card with you to meetings to speed the check in procedure.

NEWSLETTER EXCHANGE PROGRAM: We are presently exchanging newsletters with over 50 other User Groups in the USA, England, & Australia. These often contain articles, tips, & program listings that many of you would find interesting. Because of the volume, we cannot hope to get more than the best of these items into our newsletter. If you would like to sign out a file from one of these groups, please see Susan Livingston at your next meeting and she will bring it along to the following meeting. Please, one file per member per month. See the list of groups else where in this newsletter.

CASSETTE RECORDER MAINTENANCE: Bob Burley's workshop on cassette recorders has been tentatively rescheduled to the December meeting. Will advise in next newsletter.

CORRESPONDING USER'S GROUPS

As a matter of intrest. our group is exchanging newsletters with the following User Groups:

99'er LINES. NW FLORIDA USER GRP
PENSACOLA. FLORIDA. USA

ALOHA 99/4A COMPUTER USER'S GROUP
MAKAKILO. HAWAII. USA

AMARILLO 99/4 USER'S GROUP
AMARILLO. TEXAS. USA

ADELAIDE TI COMPUTER USER'S GRP.
AUSTRALIA

"BAYOU BYTE". BAYOU 99 USER GRP.
LAKE CHARLES. LA. USA

BIG SKY 99'ers COMPUTER USER GRP.
GREAT FALLS. MT. USA

T. I. BRISBANE USER GROUP
QUEENSLAND. AUSTRALIA

"BUG NEWS". BIRMINGHAM USER GRP.
TRUSSVILLE. AL. USA

FOX CITIES USER'S GROUP
APPLETON. WI. USA

BYTE LINE. DECATUR 99'er USER GRP
DECATUR. IL. USA

CEDAR VALLEY 99'er USER GROUP
MARION. IA. USA

AST TI 99'er USER'S GROUP
EDMONTON. ALTA

CHICAGO AREA TI 99/4A USER'S GRP.
CHICAGO. IL. USA

CIN-DAY USER'S GROUP
WEST CHESTER. OHIO. USA

TEXAS INSTRUMENTS INC
ATTLEBORO. MA. USA 02703

CORPUS CHRISTI TI 99/4A USER GRP.
CORPUS CHRISTI. TEXAS. USA

DALLAS TI HOME COMPUTER GROUP
IRVING. TEXAS. USA

ROCK 99 COMPUTER CLUB NEWS LETTER
WHITEWATER. WI. USA

DELAWARE VALLEY USER'S GROUP
WILMINGTON. DELAWARE. USA

THE GUILFORD 99er NEWSLETTER
GREENSBORO. NC. USA

MILWAUKEE AREA 99 4/A USER'S GRP.
WAUWATOSA. WI. USA

HOUSTON USER'S GROUP
HOUSTON. TEXAS. USA

KENTUCKIANA 99/4 COMPUTER SOCIETY
LOUISVILLE. KY. USA

broadmoor stationers ltd.

- Office Supplies
- Office Furniture
- Typewriter Rentals
- Typewriter Repairs
- Business Machines
  - Typewriters
  - Calculators
  - Cash Registers
- Photo Copy Service
- Rubber Stamps

SALES, SERVICE & LEASING

broadmoor stationers

165 Athabascan Avenue Sherwood Park

Mon. - Fri. 9:00 – 5:30 pm Sat. 10:00 - 4 p.m.

SHARP **464-4343** SCM SMITH-CORONA

KINGS 99er USER'S GROUP
HANFORD. CA. USA

MILLER'S GRAPHICS
SAN DIMAS. CA. USA

MSP 99 USERS GROUP
SAINT PAUL. MINNESOTA. USA

"MUNCH" C/O VID. CON.
WORCESTER. MA. USA

NE PENNA. AMATEUR COMPUTER CLUB
PENNSYLVANIA STATE UNIVERSITY

NET 99er GROUP
HURST. TEXAS. USA

"NHUG". NEW HAMPSHIRE 99er's
CONCORD. NH. USA

CENTRAL TEXAS 99 4/A USER'S GRP.
AUSTIN. TEXAS. USA

PITTSBURGH USER'S GROUP
PITTSBURGH. PA. USA

SAN GABRIEL VALLEY USER GROUP
W. COVINA. CA. USA

PUGET SOUND 99ers
LYNNWOOD. WA. USA

"PUNN"
PORTLAND. OR. USA

SHEBOYGAN AREA GROUP
SHEBOYGAN. WI. USA

SAVANA COMPUTER USER GROUP
SAVANA. GA. USA

SOUTHERN NEVADA USER'S GROUP
LAS VEGAS. NV. USA

"SPIRIT OF 99". CENT. OHIO 99ers
COLUMBUS. OHIO. USA

SUMMIT 99er USER'S GROUP
CUYAHOGA FALLS. OHIO. USA

"TIE-LINE". PENNANT HILLS
NEW SOUTH WALES. AUSTRALIA

THE 9900 USER'S GROUP INC.
MOORESTOWN. NJ. USA

TI BYTE
address unknown

TI HOME COMPUTER USER'S CLUB
MAIDENHEAD. BERKSHIRE ENGLAND

TIMES
BRIGHTON. SUSSEX. ENGLAND

"TIC TALK". ROCKY MOUNTAIN 99ers
LITTLETON. CO. USA

"TIDINGS"
address unknown

TRI CITIES 99er COMPUTER GROUP
WENNEWICK. WA. USA

YOUNG PEOPLES LOGO ASSC. INC.
RICHARDSON. TEXAS. USA

THE GAMES GANG

Roland PR 1111

Dot Matrix Printer.

120 c.p.s. FULL Graphics.

near letter quality mode.

1 - year warranty

THE FINEST QUALITY AVAILABLE
UNDER $1000.00

list 599.95 SALE 549.95

FINANCING AVAILABLE

Mail Orders Welcome

FAMILY COMPUTERS

REPAIRS RENTALS

435-4636

9872-63rd AVE. T6E 0G6

UPSTATE 99 4/A USER GROUP
ALBANY. NY. USA

VICTORIA 99er GROUP NEWSLETTER
VICTORIA. BC

WASHINGTON DC AREA USER GROUP
TEMPLE HILLS. MD. USA

WIREGRASS 99/4 USER'S GROUP
ENTERPRISE. AL. USA

OZARK 99er USER'S GROUP
REPUBLIC. MO. USA

"MICRO"
BLOOMINGTON. IL. USA

LEHIGH 99er COMPUTER GROUP
ALLENTOWN. PA. USA

LITTLE GEMS

Some of you readers of Home Computer Magazine (and before that. 99'er). have probably heard of Tigercub Software. This is a Mom & Pop kitchen table enterprise that is receiving good press from several newsletters published by other User's Groups. This company is run by Jim Peterson at 156 Collingwood Ave. Columbus. Ohio 43213. To quote from the Lehigh U/G newsletter:

"Jim Peterson of Tigercub Software consistently allows user groups to republish mini programs & hints that he authors. One of the on going facinations of writing a newsletter is seeing what he has come up with this month. Many of them are pretty good and all are fascinating."

You can obtain a catalog for $1.00 (refundable on your first purchase) by writing to the above address. Most orders will get you extras as Jim seems to fill up the remaining disk/cassette space with freebies.

ATTENTION MODEM USERS!

As you may have noticed already. there is an application form on the mailer sheet attached to this issue of 99'ers ON LINE. We would like to form a Sub Group (SG) of members who are interested in contacting other members via modem. Our intent here is to make our group more accessable to other members who cannot get to our meetings on a regular basis. For instance. we have several members who live some distance from town and. considering this newsletter goes to 57 other user groups. by joining this SG. you could end up being online to just about any where. This SG will be chaired by Tom Hall for the time being.

If you are interested. please fill out the form and return to the P/O box (or give to Tom at the next meeting) indicating your name. phone number. and the days and hours you would like to receive calls. Also indicate if you are a subscriber to "The Source". This list will be published in a future issue of the newsletter so if you wish us to print a "nom de plume". please indicate this on your form.

SST EXPANDED BASIC COMPILER

by: Tom Hall

One of the things you may hear discussed frequently by programmers are the pros and cons of compiled languages versus

interpretive languages. The BASIC and EXTENDED BASIC that we are all familiar with in the TI are interpreters; that is, they translate the instructions given by the user into machine language code, which is then executed by the computer. On the other hand, a compiled program is one that has been re-written in machine code and is stored in a format which can be loaded and directly executed by the computer without the need for any interpreter.

One of the supposed advantages of working with a compiled as opposed to interpreted language is speed: in theory, the compiled program, because it requires no further interpretation by the computer, will execute faster than the same set of instructions executed through an interpreter. TI's BASIC and EXTENDED BASIC systems operate through an interpreter, and althougn EXTENDED BASIC is quite a bit faster than console BASIC, it is still not as fast as machine code for this reason.

There now exists a BASIC compiler for the TI. It comes from SST Software Co. and is called the SST EXPANDED BASIC COMPILER. It allows you to write a BASIC program in much the same way as you normally would, and then convert that program into machine code, thereby taking advantage of the incredible speed of the 9900 processor. The compiler package requires 32K memory expansion, at least one disk drive, and either the EDITOR/ASSEMBLER or MINI-MEMORY command modules. Included in the package are two editor programs, two loaders and the compiler itself. With the exception of one of the editors, which can be used in EXTENDED BASIC, the entire system runs in console BASIC. As a result, the process of compiling a BASIC program is rather slow. However, your patience will be rewarded, because the speed of the compiled result has to be seen to be believed. One of the benchmark programs included with the documentation is a BASIC program which calculates prime numbers within a specified range. This program, in TI BASIC, required 4 hours 15 minutes to check the first 5500 integers, and the original program, written for the TRS-80, took 7 hours, 12 minutes to check the first 10,000 integers. The SST EXPANDED BASIC COMPILER version of this program took 11 minutes, 20 seconds to check the first 5500 integers, and only 13 seconds to check the first 1000 integers, as opposed to nearly a half-hour in console BASIC.

The process of compiling a BASIC program begins with using one of the editor programs to write your source program. If you use the EXTENDED BASIC Editor you can save your program in merged format, load the Editor, and then merge your program into the Editor. If you use the console BASIC version of the Editor, you have to load the Editor, and then type your program into the middle of the Editor program. Your line numbers can be no smaller than 11, and no larger than 32000.

Once your program is written, you resequence the Editor program (either version) with the command RESEQUENCE 1,1, and then run the program. The result is a file whicn you must use with the compiler. From this point on, you **must** be in console BASIC. The compiler loads the file created by the Editor program, and compiles it into machine language instructions, which are written out to another file. This second file is the one which you finally load and run.

One of the loader programs is designed to execute the compiled program, and another loader is designed to allow you to translate your machine code into a source file which can then be assembled with the EDITOR/ASSEMBLER into standard E/A object code.

My biggest objection to the SST EXPANDED BASIC COMPILER is it's slowness; for even the shortest BASIC program, it takes the better part of a half-hour to get your BASIC program compiled and running. Another drawback of the system is the way programs have to be entered. The system was designed with speed in mind, and in that regard is a smashing success, and in actuality the system supports almost every command available in EXTENDED BASIC. For instance, you have access to all 32 sprites (as opposed to only 28 in EXTENDED BASIC), and you can redefine any of the 256 ASCII characters instead of the limited number possible with either of the BASICs. However: you have to explicitly declare at the very beginning of your program every variable name you intend to use; you can have no more than one statement on a line; no direct string comparisons are possible, and in fact all string manipulation in the system is unwieldy -- you cannot use the DATA or READ statements; comparisons can only be made between numeric variables, and only limited types of comparisons are possible; no literal numbers can be used anywhere in the program --- all values must be referenced as variables.

However, I should point out that, even though the system is slow and cumbersome to use, it does produce a very efficient and streamlined machine code, and if you can get used to the lengthy process of writing and compiling a BASIC program using this package, you will find that the SST EXPANDED BASIC COMPILER is quite a decent piece of software, and I know of no other compiled BASIC for a microcomputer that can compare to the speed of execution of SST.

TI-WRITER: 132 COLUMN FORMATTING

by Wolly Barabash

Some problems have been raised about using the Formatter for 132 column printouts. A common mistake is the inclusion of a carriage return after each line of text. This effectively disables the 132 column mode, and leaves you in 80! When writing 132 column lines, use the space bar or keep writing using the word wrap feature to go from line to line.

Next, I haven't been able to find a simple solution to a seemingly stupid effect. That is, if you make a table, and have spaces between headings, say ten for instance, the formatter will ignore eight of these ten spaces, and pull your table in to a much smaller size! The only solution I have found is to use the transliterate command. Thus, if one uses the embedded command: ".TL 47:32", the formatter will read each / (slash) as a space (ASC equiv. of / =47, ASC equiv. of space =32). When you make your table, use the slash instead of the space bar to string words out properly. You may use other symbols instead of the slash. Use their ASC equivalents in the transliterate command. I have had this problem with an Okidata as well as another 15" carriage machine. Does anyone know of another answer?

SXB - SUPER EXTENDED BASIC

by: Tom Hall

A company in the States, called J & K H Software, has released a tidy little package called SXB, short for SUPER EXTENDED BASIC. This package is a disk-based system which auto-loads into memory over 75 machine language routines not normally available in EXTENDED BASIC. The system requires at least one disk drive and 32K memory expansion.

These enhancements fall into five major categories: data base subroutines, string array subroutines, string subroutines, integer subroutines, VDP routines, and miscellaneous.

The system's biggest strength seems to lie in its database routines. Included in these routines are facilities for sorting, updating and copying arrays, as well as rapid lookup and identification of duplicate elements. The average sorting time for a file of 200 elements ranges from 5 seconds for a file which is in almost-sorted order, to around 15 seconds for a file that is badly out of sorted order. And not only can you sort a string array, but you can specify what part of the individual element will constitute the key field for the sort; you can use multiple keys (up to 127 of them!); and you can sort in either ascending or descending order by each key. Additionally, you can count the number of items currently in an array, or determine how many items exist in a file with a key field identical to the one you specify.

In the string array subroutines are included: the facility to count the number of non-null strings in an array; determine how many bytes an array is currently using; determine the length of the longest and shortest strings in an array; translate specified characters in an array to new ones; encode a string array with a password so as to make it unreadable until the correct password is entered; and view an entire array on the screen, complete with index numbers and byte counts, with a single command -- and the array will scroll at virtually the same speed as the TI Debugger!

String subroutines include the automatic translation of decimal to binary and vice-versa; fixing the length of a string; removing specified characters or trailing spaces from a string; and swapping the value of two strings.

With the integer subroutines included in the SXB package you can "pack" 4 integers into a single numeric variable, reference each of them individually, and even do arithmetic with each or all of the four components, singly or in any combination.

The VDP routines allow you to define any part of the screen as a window, and to control the scrolling of that portion of the screen independently of the rest of the display. You can do interesting things like place a string on the screen in any one of 3 different directions (like the spokes of a wheel) with a single command. You can also redefine up to 31 consecutive character patterns with a single command, or retrive the string pattern of 31 consecutive characters in like manner. Also, a single command will turn all lower case letters into a typewriter-style character set with true descenders.

Finally, the miscellaneous subroutines enable you to create a banner on the screen with specified bits on and off, or you can direct the program to wait until a specified key is pressed (with no parameter specified it defaults to "Y/N-y/n" for "yes/no"), or you can save a parameter in one program and have it retrieved by the next program you call. In addition, there is a routine which instantly returns you to the master title screen, and there are provisions for user-written subroutines as well.

The price tag on this piece of software is a hefty $95 U.S., but as a development tool for the serious programmer, and when you consider the fact that you are getting not just a single program, but quite literally dozens, I think it's definitely a worthwhile investment.

HALF-HEIGHT DISK DRIVE INSTALLATION

by: Wolly Barabash

There are several ways dual half-heights can be installed in the peripheral expansion box. Kits are being sold by Tex-comp as well as other more "electronics" oriented firms.

The do-it yourselfer, however, can save a bit of coin if he or she follows the following instructions. Please note that the procedure should work on most expansion boxes. Also note that this was done using TEAC FD55B drives. I also used the larger screw mountings that exists on these drives. I have no reason to believe that other drives are different if they are made for the IBM. But just in case, manipulate and picture the following procedures before you do them!

A) Expansion box modifications.

1) Use the holes that exist in the expansion box in which the TI drive was set as REFERENCES.

2) With a pencil, measure from the longitudinal side of the disk drive hole, using the edge that is furthest from the card slots, a distance of 5mm. Do this for both holes.

3) Next, draw a cross hair by marking a line perpendicular (90 degree angle) to the tip of the hole facing the front of the expansion box. This line thus is in parallel with the long edge of the expansion box. Do this for both holes.

4) From the cross hair measure a distance of 4.3 cm, again away from the card slot side of expansion box, and mark it. Make new cross hairs and ensure that these are in parallel with all other parameters of the original cross hairs.(ie. the same distance away from the edge of the expansion box.)

5) You should now have four cross hairs. Two are in line with each hole.

6) Measure the complements of each cross-hair on the underside of the expansion box using the holes that exist there as references again. (Note: it will help to remove the plastic drive support in the disk drive enclosure.)

7) Drill holes using suitable bit (@1/8"). Try to extend the longitudinal dimensions of the drilled holes by wiggling the drill so as to emulate the ones made by TI.

8) Scrounge a piece of wood, such as a one inch thick plank, and cut it to approximate dimensions of the drive enclosure. This wood will replace the plastic support you took out. Tape it or hold it in place on the bottom. I drilled a hole half-way in the wood at its front right corner so as to catch the machine screw that is part of the exp. box. This way I could reposition the wood the same way every time I took it out. Next, move the expansion box so that the disk drive side is over the table edge.

Now, using a pencil, mark the spots on the underside of the wood through the new holes you drilled on the bottom of the box. Take the board out and drill these holes out using a larger bit.

9) Place washers over the holes in the wood on whose surface the drives will rest. Tape these down with masking tape and punch holes through the tape to allow the screws to go through. Important, these washers will lift the drives and allow some air circulation to go through the bottom. Put them in. The tape will prevent the drives from knocking off the washers when you install the drives.

10) I cut the power cable far back in the expansion box slots area and spliced a new set of wires so that I now had two pairs of power cords. Next I inserted the connectors for the drives A better and slightly more expensive procedure is to get a connector that hooks up to the one on the power cord, and take your two new drive connectors from that one. Thus you may merely plug it in to get two pairs.

11) If you have a cable that can hook up to two drives at once, then connect the cable to the card controller. Next unscrew one screw on the side at the back of the expansion box, and the other screws at the back of the disk enclosure. This will loosen the cabinet so that you don't bend things as you shove the drives in. You will find that it is easier to put them in both at once. Attach the cables to the drives and go ahead. Don't forget the power cords. (Note: if this two cable system is used, then take out the resistor pack on the first drive. Next ensure that drive 2 has the selector pin in D1 position. Drive 1 should have its selector pin in D0.)

12) If you use the two cable method, more modifications are needed. The expansion box disk enclosure must still be loosened as in (11) above. You must use a hacksaw, or a Dremel type model drill, and cut out the piece of metal right next to the disk drive side at the back of the expansion box. This metal has holes drilled in it by TI and extends the height of the box. Cut it out along the holes about half-way down the height of the exp. box. Next, bend it out and slip in the cable that is connected to the circuit board contacts on the disk controller. File down the sharp edges of the cut metal on both sides so that you don't cut your cable when you bend the metal piece back. Install the regular cable from the pin connector on the controller card through its usual slot. Both drives will now need resistor packs. Drive 1 will have its selector switch in D0 position, drive 2 will have its selector switch in D1 position. (note, these selector switches are on the ciruit board of the disk drive.) You are now ready to install the drives. Don't forget the power cords.

13) You are now ready to screw in the drives. Check to see that the holes are aligned, Loosely install the upper screws then tighten the bottom ones. Now tighten the top screws. Do not overtighten. You may need screws from the hardware store for these. They are the same as the ones for the TI drives, so take one along for a sample.

B) Additional work.

You may want to ensure your data integrity by placing a thin piece of sheet metal on the side of the drive which allows this (has screw holes). This will reduce magnetic interference from the drive motors etc.

I did this to both drives. I also drilled several holes in these plates over the motor plate area for circulation. The motor covering will act as a shield. Ensure that the sheet metal is thin! You may have to file down the screws a bit so that the drives can be put in without much tribulation. Close fit eh? I'm sure other modifications may be made. I hope that this will serve as a guide to those that "do".

More about half-height drives

The Teac FD55-B half-height drives can be purchased for less than those offered by Tex-Comp. See the back pages of BYTE magazine. These drives are pin for pin compatible with TI, thus no new cables need to be made. The drives are DS, DD. The TI disk manager II module will write single density, but you will be able to use two sides. The new disk controller from Cor-comp is well worth the price as it allows double density storage. Imagine your diskette library increasing in value by at least half, thus offsetting the purchase price of the drives and card controller. There is always a risk with warranty, however, when dealing with the U.S.

TRANSFERING COMPANION FILES VIA TE II

By Tom Hall

A short while back I had a slight problem involving writing articles for the newsletter. The only word processing system I have is COMPANION, and Bob Pass produces this newsletter with TI-WRITER. I discovered that, some time ago when I submitted an article which Bob published, he had actually printed out the COMPANION file I gave him, and then re-typed the article on his TI-WRITER so that it could be formatted in the news letter style. We both agreed that the duplication of typing should be unnecessary, so, with the help of Michal Jaegermann, we finally discovered a workable solution to the problem. In general, this solution can be applied to a number of situations, but I'll just describe the specifics of this particular one.

The person who is sending the file should load COMPANION (or whatever system he uses), and prepare to print the file just as he would if he were going to send it to his printer -- the only exception being that instead of printing the file to a hard copy device, the file is going to be "printed" to the modem! You can experiment with printer statements, but the simple "RS232" with no modifiers will probably work. With COMPANION you have the option of specifying single-sheet print mode, in which case COMPANION will wait for you to put a new sheet of paper in your printer and then resume printing. This single sheet mode was necessary for our application, since Bob was going to receive the printed file on his TV screen, and would have to capture it using the CTRL 2 (OUTPUT) option on the TERMINAL EMULATOR II. When receiving (or "downloading") a file in this manner, you should make sure that you have turned off Word Wrap (CTRL 5), so as to maximize the amount of information you can capture at one time. The screen buffer on the TE II in this mode is approximately 1 1/2 screens-full, or about 36 lines.

With COMPANION, I had to specify a page size of 33, which is the smallest setting allowed with that system, and from here on out things got a bit tricky. We put our phones in our modems, and as soon as I saw my ready light come on, I started printing the file to the modem. After 33 lines, COMPANION stops and waits for a signal to continue.

At this point, things on the receiving end get busy. As soon as your screen stops scrolling, you should use the CTRL 2 option to output what is in your screen buffer to a disk file. While the system is writing the buffer to disk, a message will appear on your screen, telling you to please wait. As soon as the buffer is written, whatever was on your screen will re-appear. AT THIS POINT YOU SHOULD COMPLETELY EMPTY YOUR BUFFER BY TYPING CTRL 5 TWICE. At this point, if you are on the receiving end, you need some way of letting the sender know you are ready to receive again. Bob and I hit upon the idea of briefly turning the modem off and on again. In most instances, the sender's modem will either have a ready lamp, or else will make some kind of noise to indicate that something has happened, at which time printing can be re-started, and the whole process repeats.

Some practice may be necessary before you completely get the hang of this technique, but, although somewhat cumbersome, it does work!!

Editor's note: This whole rigamarole is neccessary because of the incompatible COMPANION file structure with TI Writer. So, if any of you out there have COMPANION (or TI Writer) and a modem, you can now send me your articals for the news letter directly instead of by mail!